just completed copying. For example, if the snapshot worker **206** informs the coordinator **216** that database tables "1", "5", and "8" in database **204** have just been respectively copied to the flat files **222**, **224** and **226**, the coordinator **216** determines whether the delete workers (**242–248**) have deleted snapshot tables "1", "5" and "8" in snapshot table database **240**, thereby removing the "stale data".

In certain embodiments, the coordinator **216** uses the information in the copy table list **218** to determine if a particular snapshot table has been deleted. In one embodiment, the coordinator **216** uses the delete complete time **254** parameter to determine if a snapshot table has been deleted.

If at step **520** the coordinator **216** determines that a snapshot table has not been deleted (i.e. snapshot tables "1", "5" and "8" for the previous example), then at step **522**, the coordinator **216** delays the launching (e.g. spawning) of a loader worker (**230–238**) for loading the information until the snapshot tables "1", "5" and "8" have been deleted. For example, assuming that snapshot worker **206** has notified coordinator **216** that it has copied database tables "1", "5" and "8" from database **204** into flat file **220**, if coordinator **216** determines that snapshot table "5" in snapshot table database **240** has not been deleted (i.e. the "stale data" has not been removed), then coordinator **216** will delay launching loader worker **230** for loading the information into snapshot tables "1", "5" and "8", until snapshot table "5" has been deleted. Control then proceeds to step **534**.

If at step **520**, the coordinator **216** determines that the snapshot-table has been deleted, then at step **524** the coordinator **216** launches a loader worker (**230–238**) to load the information from the flat file into the corresponding snapshot table in snapshot table database **240**.

At step **526**, when a loader worker (**230–238**) finishes loading a flat file (**220–228**) into its corresponding snapshot file in snapshot table database **240**, it notifies the coordinator **216**. In certain embodiments, the loader workers (**230–238**) notify the coordinator **216** of the particular snapshot table in which the data was loaded.

At step **528**, the coordinator **216** determines whether all the desired tables identified in copy table list **218** have been copied from database **204** into the snapshot tables in snapshot table database **240**.

If at step **528** the coordinator **216** determines that all the tables identified in copy table list **218** have been copied, then at step **530**, the coordinator **216** notifies the memory based planner **202** that the snapshot table database **240** contains a valid copy of the desired tables from database **204**. At step **532**, the memory based planner **202** uses the snapshot tables in snapshot table database **240** to generate the planning schedule.

Conversely, if at step **528** the coordinator **216** determines that all the tables identified in copy table list **218** have not been copied, then at step **534**, the coordinator **216** continues to accept and process other completion notifications from snapshot workers (**206–214**). When the coordinator **216** receives a notification that a snapshot worker (**206–214**) has finished copying its assigned data to a particular flat file, control proceeds to step **520** to determine whether the delete workers (**242–248**) have deleted the corresponding snapshot tables in the snapshot table database **240**.

Copying Data Directly Into Another Database

Certain database systems provide snapshot mechanisms for transactions that modify the database, as well as for "read only" transactions. Such a mechanism is described in U.S. patent application Ser. No. 08/613,026, filed Mar. 11, 1996, Entitled "Method and Apparatus for Providing Isolation Levels in a Data", the contents of which are incorporated herein by this reference. In certain embodiments, such a snapshot mechanism is used to allow the snapshot workers to copy the desired tables into a database without the use of flat files.

FIG. **6** depicts a block diagram of a system **600** that is used for producing a copy of a database in accordance with an embodiment of the present invention. FIG. **6** is similar to FIG. **2**, and therefore like components have been numbered alike.

As illustrated in FIG. **6**, snapshot workers (**206–214**) retrieve database table information from database **204** and copy it directly into the snapshot table database **240**. By writing directly to the snapshot table database **240**, the intermediate steps of copying data to flat files and launching loader workers to copy that data from the flat files into the snapshot table database **240** can be eliminated. In one embodiment, the snapshot workers (**206–214**) copy the database tables (1..N) in database **204** directly into temporary files in snapshot table database **240**. These temporary files are used by the memory based planner **202** for generating a planning schedule.

In an alternative embodiment, the snapshot workers (**206–214**) copy the desired database tables (1..N) in database **204** directly into snapshot table database **240** as "large binary objects" (BLOBs). In general, a BLOB is a data item that consists of a large amount of data and may be used by systems that require data types that are typically much larger than traditional data types. For example, a single BLOB may include four gigabytes of data and may be thought of as a file or a stream of characters or bytes.

In certain embodiments, snapshot workers (**206–214**) retrieve database table information from database **204** and copy it as separate data back into database **204**. By copying the database table information from database **204** as separate data in database **204**, the intermediate steps of copying the data to flat files can be eliminated. In one embodiment, the desired database tables (1..N) in database **204** are directly copied as separate BLOBs in database **204**.

In the foregoing specification, the invention has been described with reference to specific embodiments thereof. It will, however, be evident that various modifications and changes may be made thereto without departing from the broader spirit and scope of the invention. The specification and drawings are, accordingly, to be regarded in an illustrative rather than a restrictive sense.

What is claimed is:

1. A method for supplying a consistent set of data to a software application, the method comprising the steps of:

launching said software application;

identifying a particular set of data that is required by the software application;

requesting a first process to obtain a snapshot time from a database server associated with a first database, wherein the snapshot time causes all subsequent reads of said first database by the first process to return data that reflects a database state associated with the snapshot time;

after the first process obtains the snapshot time, causing the first process to extract the particular set of data from the first database; and

supplying said software application with the particular set of data that was extracted from the first database.

2. The method of claim 1, further comprising the step of causing a second process to store the particular set of data in a second database.

3. The method of claim 2, wherein the step of causing the second process to store the particular set of data in the second database includes the steps of:

writing the particular set of data to one or more flat files; and

executing a loader process, wherein the loader process loads the particular set of data from the one or more flat files to the second database.

4. The method of claim 3, wherein the step of writing the particular set of data to one or more flat files includes the steps of:

the first process informing a coordinator process when it has finished writing data to a particular flat file; and

the coordinator using the information to tell the loader process when it can begin loading the flat file into the second database.

5. The method of claim 3, wherein the step of:

writing the particular set of data to the flat file includes the step of writing the particular set of data to a plurality of flat files; and

executing the loader process includes the step of executing a plurality of loader processes, wherein the plurality of loader processes load the particular set of data from the plurality of flat files to the second database.

6. The method of claim 2, wherein the step of supplying said software application with data from said particular set of data includes the steps of:

said software application reading the particular set of data stored in the second database; and

said software application generating a planning schedule based on the particular set of data.

7. The method of claim 1, wherein the step of identifying the particular set of data includes the step of creating a copy table list, wherein the copy table list contains entries that identify the particular set of data in the first database.

8. The method of claim 7, further comprising the steps of:

executing a delete process, wherein the delete process uses the copy table list to identify data that needs to be deleted in a second database; and

deleting the identified data from the second database.

9. The method of claim 7, where the step of creating the copy table list includes the steps of:

communicating with the software application to identifying a set of planning data, where the planning data is required for generating a planning schedule; and

creating the copy table list based on the identified set of planning data.

10. The method of claim 1, wherein the step of supplying said software application with data from said particular set of data includes the steps of:

writing the particular set of data to one or more flat files; and

supplying the one or more flat files to said software application, wherein said software application generates a planning schedule based on information contained in the one or more flat files.

11. A method for producing a copy of data from a first database, the method comprising the steps of:

locking a first set of data in the first database;

after locking the first set of data,

requesting a plurality of processes to obtain snapshot times from a database server associated with said first database, wherein the snapshot times cause all subsequent reads of the first database by the plurality of processes to return data from the first database as of said snapshot times;

waiting a particular period of time for the plurality of processes to be assigned snapshot times;

releasing the locks on the first set of data in the first database;

using a successful set of said plurality of processes to extract a copy of the first set of data from the first database, wherein said successful set includes only those processes of the plurality of processes that were assigned a snapshot time within the particular period of time; and

storing the copy of the first set of data separate from said first of data.

12. The method of claim 11, wherein the step of identifying the first set of data includes the step of creating a copy table list, wherein the copy table list contains entries that identify the first set of data in the first database.

13. The method of claim 12, where the step of creating the copy table list includes the steps of:

identifying a set of planning data, where the planning data is required to generate a planning schedule; and

creating the copy table list based on the planning data required to generate the planning schedule.

14. The method of claim 12, further comprising the steps of:

executing a plurality of delete processes, wherein the plurality of delete processes use the copy table list to identify data that needs to be deleted in a second database; and

deleting the identified data from the second database.

15. The method of claim 11, wherein the step of storing the copy of the first set of data includes the steps of:

writing the copy of the first set of data to a plurality of flat files; and

executing a plurality of loader processes, wherein the plurality of loader processes load the copy of the first set of data from the plurality of flat files to a second database.

16. The method of claim 15, wherein:

the steps of writing the copy of the first set of data to a plurality of flat files further includes the step of notifying a coordinator process that data has been written to one of the plurality of flat files; and

the steps of executing the plurality of loader processes further includes the step of the coordinator, upon being notified that data has been written to one of the plurality of flat files, launching a loader process to load the first set of data from one of the plurality of flat files to the second database.

17. The method of claim 15, wherein the step of writing the copy of the first set of data to the plurality of flat files includes the steps of:

the plurality of process informing a coordinator process when it has finished writing data to a particular flat file; and

the coordinator using the information to tell one of the plurality of loader processes when it can begin loading the particular flat file into the second database.

18. The method of claim 11, wherein the step of requesting the plurality of processes to obtain a snapshot time includes the step of requesting the plurality of processes based on a user input parameter, wherein the user input

parameter identifies how many processes should be requested to obtain a snapshot time.

19. The method of claim 11, wherein the step of extracting the copy of the first set of data from the database includes the steps of:

assigning a set of copy data to the plurality of snapshot processes; and

retrieving data from the first database based on the set of copy data that was assigned to the plurality of snapshot processes.

20. The method of claim 11, wherein the step of storing the copy of the first set of data includes the steps of storing the copy of the first set of data as blob files that are separate from said first of data.

21. The method of claim 11, wherein the step of storing the copy of the first set of data includes the steps of storing the copy of the first set of data in said first of data.

22. A computer-readable medium carrying one or more sequences of one or more instructions for supplying a consistent set of data to a software application, the one or more sequences of one or more instructions including instructions which, when executed by one or more processors, cause the one or more processors to perform the steps of:

launching said software application;

identifying a particular set of data that is required by the software application;

requesting a first process to obtain a snapshot time from a database server associated with a first database, wherein the snapshot time causes all subsequent reads of said first database by the first process to return data that reflects a database state associated with the snapshot time;

after the first process obtains the snapshot time, causing the first process to extract the particular set of data from the first database; and

supplying said software application with the particular set of data that was extracted from the first database.

23. The computer-readable medium of claim 22, wherein the computer-readable medium further comprises instructions for performing the step of causing a second process to store the particular set of data in a second database.

24. The computer-readable medium of claim 23, wherein the step of causing the second process to store the particular set of data in the second database includes the steps of:

writing the particular set of data to one or more flat files; and

executing a loader process, wherein the loader process loads the particular set of data from the one or more flat files to the second database.

25. The computer-readable medium of claim 24, wherein the step of writing the particular set of data to one or more flat files includes the steps of:

the first process informing a coordinator process when it has finished writing data to a particular flat file; and

the coordinator using the information to tell the loader process when it can begin loading the flat file into the second database.

26. The computer-readable medium of claim 22, wherein the step of identifying the particular set of data includes the step of creating a copy table list, wherein the copy table list contains entries that identify the particular set of data in the first database.

27. A computer-readable medium carrying one or more sequences of one or more instructions for producing a copy of data from a first database, the one or more sequences of one or more instructions including instructions which, when executed by one or more processors, cause the one or more processors to perform the steps of:

locking a first set of data in the first database;

after locking the first set of data,

requesting a plurality of processes to obtain snapshot times from a database server associated with said first database, wherein the snapshot times cause all subsequent reads of the first database by the plurality of processes to return data from the first database as of said snapshot times;

waiting a particular period of time for the plurality of processes to be assigned snapshot times;

releasing the locks on the first set of data in the first database;

using a successful set of said plurality of processes to extract a copy of the first set of data from the first database, wherein said successful set includes only those processes of the plurality of processes that were assigned a snapshot time within the particular period of time; and

storing the copy of the first set of data separate from said first of data.

28. The computer-readable medium of claim 27, wherein the step of identifying the first set of data includes the step of creating a copy table list, wherein the copy table list contains entries that identify the first set of data in the first database.

29. The computer-readable medium of claim 27, wherein the step of storing the copy of the first set of data includes the steps of:

writing the copy of the first set of data to a plurality of flat files; and

executing a plurality of loader processes, wherein the plurality of loader processes load the copy of the first set of data from the plurality of flat files to a second database.

30. The computer-readable medium of claim 29, wherein the step of writing the copy of the first set of data to the plurality of flat files includes the steps of:

the plurality of process informing a coordinator process when it has finished writing data to a particular flat file; and

the coordinator using the information to tell one of the plurality of loader processes when it can begin loading the particular flat file into the second database.

31. The computer-readable medium of claim 27, wherein the step of extracting the copy of the first set of data from the database includes the steps of:

assigning a set of copy data to the plurality of snapshot processes; and

retrieving data from the first database based on the set of copy data that was assigned to the plurality of snapshot processes.

32. A computer system for supplying a consistent set of data to a software application, the computer system comprising:

a memory;

one or more processors coupled to the memory; and

a set of computer instructions contained in the memory, the set of computer instructions including computer instructions which when executed by the one or more processors, cause the one or more processors to perform the steps of:

launching said software application;

identifying a particular set of data that is required by the software application;

requesting a first process to obtain a snapshot time from a database server associated with a first database, wherein the snapshot time causes all subsequent reads of said first database by the first process to return data that reflects a database state associated with the snapshot time;

after the first process obtains the snapshot time, causing the first process to extract the particular set of data from the first database; and

supplying said software application with the particular set of data that was extracted from the first database.

33. The computer system of claim **32**, further including instructions for performing the step of causing a second process to store the particular set of data in a second database.

34. The computer system of claim **33**, wherein the step of causing the second process to store the particular set of data in the second database includes the steps of:

writing the particular set of data to one or more flat files; and

executing a loader process, wherein the loader process loads the particular set of data from the one or more flat files to the second database.

35. The computer system of claim **34**, wherein the step of writing the particular set of data to one or more flat files includes the steps of:

the first process informing a coordinator process when it has finished writing data to a particular flat file; and

the coordinator using the information to tell the loader process when it can begin loading the flat file into the second database.

36. The computer system of claim **32**, wherein the step of identifying the particular set of data includes the step of creating a copy table list, wherein the copy table list contains entries that identify the particular set of data in the first database.

37. A computer system for producing a copy of data from a first database, the computer system comprising:

a memory;

one or more processors coupled to the memory; and

a set of computer instructions contained in the memory, the set of computer instructions including, computer instructions which when executed by the one or more processors, cause the one or more processors to perform the steps of:

locking a first set of data in the first database;

after locking the first set of data,

requesting a plurality of processes to obtain snapshot times from a database server associated with said first database, wherein the snapshot times cause all subsequent reads of the first database by the plurality of processes to return data from the first database as of said snapshot times;

waiting a particular period of time for the plurality of processes to be assigned snapshot times;

releasing the locks on the first set of data in the first database;

using a successful set of said plurality of processes to extract a copy of the first set of data from the first database, wherein said successful set includes only those processes of the plurality of processes that were assigned a snapshot time within the particular period of time; and

storing the copy of the first set of data separate from said first of data.

38. The computer system of claim **37**, wherein the step of identifying the first set of data includes the step of creating a copy table list, wherein the copy table list contains entries that identify the first set of data in the first database.

39. The computer system of claim **37**, wherein the step of storing the copy of the first set of data includes the steps of:

writing the copy of the first set of data to a plurality of flat files; and

executing a plurality of loader processes, wherein the plurality of loader processes load the copy of the first set of data from the plurality of flat files to a second database.

40. The computer system of claim **39**, wherein the step of writing the copy of the first set of data to the plurality of flat files includes the steps of:

the plurality of process informing a coordinator process when it has finished writing data to a particular flat file; and

the coordinator using the information to tell one of the plurality of loader processes when it can begin loading the particular flat file into the second database.

41. The computer system of claim **37**, wherein the step of extracting the copy of the first set of data from the database includes the steps of:

assigning a set of copy data to the plurality of snapshot processes; and

retrieving data from the first database based on the set of copy data that was assigned to the plurality of snapshot processes.

* * * * *